TRU Student Class Combinations and Reddit Comment Data Analysis

Evan Mackay

Kristofer Campbell

Introduction

For our final project, we decided on two datasets that we found to be interesting in the sense that they were both relevant to the course material in this class, and they were relevant to our interests as well.

The first dataset is a list of all TRU student's class combinations for Winter 2018. We were able to obtain this data simply by asking the TRU IT department nicely. They were kind enough to provide the data by writing an SQL statement to return the necessary data, while filtering out unnecessary information such as students that dropped the classes, or only audited the class (Code Block A). In addition to simply being an interesting dataset to analyze, we also believed that our work could possibly help TRU with eliminating some course conflicts in the future. Currently, TRU predicts the number of students that will take a class simply by using the number of students that took the class in the previous semester, but they don't take into account the growth or decline of a field of study. In conjunction with this dataset we used the publicly available course data that contains information about the course, location, meeting times, and the professor teaching the course.

The second dataset is of all Reddit comments in the month of January of 2015 (Baumgartner). This dataset was particularly challenging to work with, simply because of its size. This dataset contained all usernames, subreddit information, time the comment was posted, number of upvotes, number of downvotes and much more, for an entire month of one of the internet's largest websites. Originally, the entire dataset was

larger than 30 GB. After working with it and removing the information we did not need, one of our final CSV files for one of our analysis methods was a little more than 8 MB.

TRU Class Associations

One of the methods we used to analyse the TRU student dataset was to apply association rules in order to determine classes that are often taken together in a single semester.

Our first attempt at mining association rules proved unsuccessful. We only used a single semester of data and we used every course on the list. This was unsuccessful for two reasons. Firstly, the diversity of the itemsets in the dataset was quite high which required extremely low cutoffs for confidence and lift. Secondly, the dataset was not filtered at all which allowed what I called ‘prescribed semesters’ to massively influence the association rules. ‘Prescribed semesters’ is the name I gave to semesters in which the student is not given the option to choose which classes to take. This skews the results when a large amount of students take ‘prescribed semesters’ the classes in those semesters become associated with other classes in the ‘prescribed semester.’

Once we became aware of the problems we started filtering the dataset. The first thing we did was remove the ‘prescribed semesters’, we discovered that certain subjects have ‘prescribed semesters’ so we were able to add a filter to disinclude the subjects that we knew had ‘prescribed semesters.’ To solve the issue of the diversity of the itemsets being too high we both choose to find association rules of class combinations that contained a specified subject and decided to use 7 years of class

combinations instead of a single semester. We chose the subject ‘COMP’ as we were most familiar with it. This provided interesting results, when we sorted the ‘COMP’ rules by confidence we found that the top 9 rules didn’t contain a single ‘COMP’ class, instead contained ‘BIOL’ classes being associated with other ‘BIOL’ classes. We theorized that this was due to Biology being a much more popular subject than Computer Science. Combinations containing COMP were 70% as common as combinations containing BIOL. By looking at the program requirements we were able to see that Biology students are encouraged to take certain Computer Science classes as electives. Both of these theories would explain the top association rules for Computer Science. We will look into whether this appears with other subjects as a way of optimizing how TRU could schedule classes.

TRU Class Size vs Length

We noticed at TRU in our experience there was very small class sizes. We wanted to see if this expanded across different subjects and if so, does class length have an impact. We saw (Figure A7 the x axis being class size and the y axis being class length) that a couple clusters emerged. One cluster of the orange points represented the nursing practicum while the business classes had large class sizes with relatively short class length. We also noticed how the majority of the classes taught at TRU had under 80 students with the size sharply declining as class length increased. We theorized that this had more to do with TRU scheduling than it had to do with students preferring to take smaller class sizes. There would be no way of knowing as

currently students choose classes from what the schedule shows. There is also very little option to take classes of different length at TRU due to limited options.

Subreddit Associations

From the original Reddit dataset (Baumgartner), only two variables were needed from each data point to create associate rules for subreddits.

Firstly, the username of the user that posted the comment. This will help us with grouping the data into a list of subreddits that each user participates in. Secondly, the subreddit name where the comment was posted. With these two variables for each data point, it is possible to create a list of all subreddits that each user commented on in January of 2015.

One problem with our data gathering method in this case, is that user participation on Reddit should not completely be linked to commenting. There are many more ways to participate on a subreddit other than commenting on a post. Users can vote on posts or comments, or submit their own content. These forms of activity were not included in our dataset, so any association rules for user activity on a subreddit is solely linked to which subreddits they commented on in January of 2015.

Another potential problem is that subreddits which essentially represent the same concept may have their associations disproportionately changed. If a user participates on r/NBA for example, they may not participate on r/Basketball, even though it is something they may be interested in, simply because they have satisfied their basketball content by already participating in r/NBA.

Our hope for this data analysis was to find interesting relationship between unrelated subreddits. For example, if there was a strong correlation between r/Cats and r/Bikes, that would be a surprising result that we would be interested in. We were less interested in relationships between obviously related subreddits. For example a relationship between r/Cats and r/Kittens would be unsurprising and mostly uninteresting.

We expected default subreddits to dominate the top of the association rules. Not just because they would have the most participation, but because new Reddit users may be more likely to be stuck participating in the same default subreddits. So on average, a default subreddit may be more likely to be associated with other default subreddits only because users may not know where to find other subreddits.

Our actual findings determined that the default subreddits in fact did not dominate the top of the association rules. The top rule when sorting by lift was r/Nintendo → r/3DS (Figure B2). Like mentioned before, association rules like this are rather uninteresting because they are quite predictable. Unfortunately, we didn't predict that since the range of data is so large, because there are thousands of subreddits, that it is practically impossible to find interesting relations between seemingly unrelated subreddits. Instead of being dominated by the default subreddits, our association rules were instead populated by predictable combinations of sports subreddits or other related topics. We looked for a long time for any interesting combination of subreddits in our association rules, but failed to find anything noteworthy. Perhaps given a longer

time period than one month, users would be more likely to participate in a range of subreddits, and that would give more interesting results.

Subreddit Hourly Activity Bagging

From the original huge Reddit dataset with all information that can possibly be obtained from a Reddit comment for all of January 2015 (Baumgartner), only three variables were needed from each piece of data in order to classify a subreddit into how active it was in each hour of the day.

Firstly, the date and time a comment was posted. The original time given from the dataset was in a unix timestamp format, which means that we had to convert it to a normal date-time format. After doing this, and converting everything into UTC time (Code Block D), all the comments ranged from January 1 00:00:00 to January 31 23:59:59.

Secondly, the subreddit name that the comment was posted to. This variable is how we later categorized the data. It is technically possible to categorize the activity of a subreddit using only these two variables, but in order to get a more accurate representation, we also included a third.

Lastly, the total votes a comment received. A comment on Reddit can either be upvoted (+1) or downvoted (-1). The total vote count on a comment is calculated by taking the number upvotes and subtracting the number of downvotes. By taking into consideration the total vote count of a comment, we can better quantify the activity of a

subreddit at a certain time. A comment can be voted on hours after it was posted of course, but on average we believe this helps to categorize user activity.

After all the comment data was shortened to its timestamp, subreddit and vote count (Code Blocks C and D), it was now possible to combine all the data points that shared the same subreddit, and then further categorized it into the hour of the day it was posted. Essentially, each subreddit had 24 values for the total count of votes on comments that were posted in their respective hour. Since all timestamps were in UTC time, the hours are standardized so it is then possible to compare subreddit activity throughout a 24 hour period (Code Block E).

Many comparisons between most subreddits would be pointless, such as comparing r/Cats and r/Tea would be mostly meaningless because the subreddits are unrelated, and any difference in peak hours would most likely be random. However, there are interesting comparisons that can be made between subreddits such as r/Cats and r/Dogs for example. If there was a large difference in the peak of related subreddits, it could suggest the the average user on one subreddit is more likely to be active at a different time of day compared to the other subreddit. Our hopes were to find interesting comparisons between related subreddits, but we also expected to find a general user activity pattern that would apply to almost all subreddits. It would make sense to find this, because the participation of a subreddit and its related topic would likely have very minimal effect on the overall schedule of the user.

After adding some of the top subreddits to a plot of time vs votes, and ranking the data so red represented subreddits with the highest overall participation, and blue had

the lowest (Figure A1), it was clear that the top subreddit in terms of votes on comments was r/AskReddit. AskReddit has so much participation in fact, that it distorted the rest of the chart and made patterns harder to see. After flipping the axis, and removing r/AskReddit from the data, a clear pattern emerged from the data (Figure A2). There appears to be a very strong correlation between the time a comment is posted, and the number of votes the comment gets. For example, in the largest subreddits, a comment posted at 9am UTC is expected to have approximately a quarter the number of total votes as a comment at 6pm UTC. A linear regression line is added to show the overall trend of these subreddits (Figure A3). In Figure A4, some of the subreddits were removed to increase the readability, and the data was bagged.

The Bagging or Bootstrap Aggregation was done by first bootstrapping the average of a subreddit at each individual hour. This was done for each hour and plotted as a grey line on the graph (Figure A4 and A5). This was repeated 50 times (50 was an arbitrary number we found with trial and error). We then bootstrapped the values of all of the previously bootstrapped values at each hour to find the final bagged line. This was plotted in black (Figure A4 and A5).

Our code allowed for easy analysis of a given list of subreddits, For example we graphed all subreddits containing the word ‘cats’ (Figure A6) As in the situation with the most popular subreddits we had to exclude the most popular subreddit ‘cats’ as it was warping the scale of the graph. Unfortunately our dataset predates our favourite cat related subreddit ‘SiberianCats’. There was too few comments across cat related subreddits to produce a trend.

We also wanted to show the potential danger of the result if a subreddit was associated with a certain geographical location. Of course, if a subreddit is extremely particular to a location, the activity on the subreddit would correlate with that location's sleeping and activity patterns. In order to demonstrate this, we took some of the largest location subreddits and compared them to each other. Most of the subreddits belong in North America, and there is little variation among those, but a few European and Oceanic subreddits are included which are in green shades. It is clear that those subreddits have a drastically different activity pattern. Even among the North American subreddits, there is a clear shift in activity hours between r/Toronto and r/Vancouver. Vancouver appears to be a few hours later than Toronto, which makes sense since the West coast is 3 hours behind the East coast time zone. This data was also bagged, and the bagging lines are displayed.

References

Baumgartner, J. (n.d.). Learn about Big Data and Social Media Ingest and Analysis. Retrieved from https://pushshift.io/

Eric Youd (2018, March 26) Person Conversation and datasource

L. M. Sheikh, B. Tanveer and M. A. Hamdani, "Interesting measures for mining association rules," *8th International Multitopic Conference, 2004. Proceedings of INMIC 2004.*, 2004, pp. 641-644.

Appendix

Code Block A (SQL)

```
        select   student_uid      as  "UNIQUEID",  --Delete UNIQUEID in output to anonymize

        student_term_code       as  "TERM",
       listagg(sections_subj_code||sections_crse_numb||sections_seq_numb) within group (order by
sections_subj_code,sections_crse_numb,sections_seq_numb)  as  "COURSES"
from    studentDB.student join studentDB.sections
                on sections_term_code = student_term_code
                and sections_crn          = student_crn
-----------------------------------------------------------------
where   student_campus_code = 'Kamloops'                --Campus Only
  and   student_term_code like '%0'               --No OL semesters
  and   student_type_code not like 'D%'             --Ignore Drops
  and   student_course_code not in ('Audit','Cancel')  --No Audit/Cancels
  and   sections_type_code <> 'N'                   --No Sem/Labs
  and   sections_subj_code not like 'X%'            --No Non-Transcriptables
group by        UNIQUEID,
        student_term_code
order by        "TERM" desc,
        "COURSES"
```

Code Block B (Python)

```
# Evan Mackay
#
import csv
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.cm as cm
from random import random,choice
import numpy as np


header = ['subreddit',0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
def format():
	output = []
	header = list(map(str,header))
	output.append(header)
	with open('newnewsubredditData.csv','r') as file:
		reader = csv.reader(file,delimiter=',',quotechar='"')
		for line in reader:
			if line:
				temp_dict = eval(line[1])
				counts = list(temp_dict.values())
				counts = list(map(str,counts))
				output.append([line[0]]+counts)

	with open('formatedsubtimes.csv','w') as out:
		for line in output:
			out.write(','.join(line)+'\n')

def bootstrap(dataset:list) -> list:
	outset = []
	while len(outset) != len(dataset):
		outset.append(int(choice(dataset)))
	return outset

subs_to_graph = {
		'topcities':[
			'nyc', 'seattle', 'chicago', 'toronto', 'losangeles',
			'portland', 'boston', 'austin', 'london', 'sanfrancisco',
			'vancouver', 'washingtondc', 'houston', 'atlanta', 'philadelphia',
			'denver', 'melbourne', 'sandiego', 'dallas', 'montreal', 'sydney',
			'pittsburgh', 'calgary', 'baltimore', 'stlouis' #36ish
			],
		'topsubs':[
			'AdviceAnimals', 'WTF', 'funny', 'explainlikeimfive', 'TumblrInAction',
			 'news', 'pics', 'worldnews', 'movies', 'tifu', 'todayilearned',
			'leagueoflegends', 'DestinyTheGame', 'nfl', 'videos', 'gifs', 'soccer',
			'DotA2', 'pcmasterrace', 'gaming', 'anime', 'KotakuInAction', 'SquaredCircle',
```

```
                        'relationships', 'GlobalOffensive'#,'AskReddit' #askreddit has way more
comments than others, throws off scale
                                ],
                'europe': [
                        'Europe','UnitedKingdom','Sweden','TheNetherlands','Ireland','France','London',
                        'Denmark','Italy','Norge','Suomi','Germany','Polska','DE','Belgium','România',
                        'Scotland','Austria','Russia','Greece','Ukraina','Norway','Iceland','Croatia','Portugal'
                        ]
                }

x = []
y = [header[1:]]

with open('formatedsubtimes.csv','r') as file:
        dreader = csv.reader(file)
        for line in dreader:
                nums = list(map(int,line[1:]))
                name = line[0]

                min_ = min(nums)
                max_ = max(nums)

                #for max of each subreddit
                #x.append(line.index(str(max_)) - 1 + ( random() * 0.1) )
                #y.append(max_)

                #for each subreddit need to iterate once to get total num of subs
                if name.upper() in (name.upper() for name in subs_to_graph['europe']):
                        print(name)
                        x.append(line)
colors = cm.rainbow(np.linspace(0, 1, len(x)+1))

#so red is more popular
colors = colors[::-1]
x = sorted(x,reverse=True,key=lambda a: max(list(map(int,a[1:]))))

fig = plt.figure()
ax1 = fig.add_subplot(111)

legends = []

for count,votes in enumerate(x):

        nums = list(map(int,votes[1:]))#cause csv is read as str need to cast
        ax1.scatter(y,nums,c=colors[count]) #graph with colours specified earlier
        #print(votes[0],votes[1:]) # logging
        #print(nums,y[0])
        #print(np.polyfit(nums,y[0],2))

        legend = mpatches.Patch(color=colors[count], label=votes[0]) #label
        legends.append(legend)
        #.plot(nums, np.poly1d(np.polyfit(x, y, 1))(np.unique(x)))

#my attempt at bagging
```

```
all_bag = []
for _ in range(50):
	bag_list = []
	for hour in y[0]:
		hour_list = []
		bootstrap_hour = []
		for sub in x:
			hour_list.append(sub[1:][hour])
		for _ in hour_list:
			boot = bootstrap(hour_list)
			bootstrap_hour.append(np.mean(boot))
		bag_list.append(np.mean(bootstrap_hour))
	plt.plot(y[0],bag_list,color='grey',alpha=0.3)
	all_bag.append(bag_list)

#second iteration
final = []
for hour in y[0]:
	bag_list = []
	hour_list = []
	for bag in all_bag:
		bag_list.append(bag[hour])
	for _ in bag_list:
		boot = bootstrap(bag_list)
		hour_list.append(np.mean(boot))
	final.append(np.mean(bootstrap(hour_list)))

plt.plot(y[0],final,color='black')
plt.xticks(np.arange(0, 24, 1.0))
plt.legend(handles=legends)
plt.show()
```

Code Block C (Python)

```
import pickle
import json
from datetime import datetime as dt

i = 0
all_ = []
with open('RC_2015-01','rb') as file:
    for line in file:
        j = json.loads(line)
        all_.append([j['created_utc'],j['subreddit'],str(j['score'])])
        if i % 100000 == 0: print(i)
        i+=1

with open('kris_list.csv','w') as out:
    for line in all_:
        out.write(','.join(line)+'\n')
```

Code Block D (Python)

```
newall = []
i = 0
with open('kris_list.csv','r') as csv_file:
    for line in csv_file:

        line_list = line.split(',')
        line_list[-1] = line_list[-1].replace('\n','') #get ride of newline char

        hour = str((dt.fromtimestamp(int(line_list[0])).hour + 8)%24)

        line_list[0] = hour
        newall.append(line_list)
        if i%100000 == 0: print(i)
        i += 1


with open('3new_list.csv','w') as out:
    for line in newall:
        out.write(','.join(line)+'\n')
```

Code Block E (Python)

```
import csv
subredditHourDictionary = {}
newHourDict =
{0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0,15:0,16:0,17:0,18:0,19:0,20:0,21:0,22:0,2
3:0}
i = 0

with open('3new_list.csv','r') as csv_file:
     for line in csv_file:
        hourPosted = int(line.split(",")[0])
        subredditName = line.split(",")[1]
        totalVotes = int(line.split(",")[2])

        #New subreddit
        if subredditName not in subredditHourDictionary:
            subredditHourDictionary[subredditName] = newHourDict.copy()

        #Adding votes to correct hour of dictionary
        subredditHourDictionary[subredditName][hourPosted] += totalVotes

        if i %100000 == 0: print(i)
        i+=1

with open('newnewsubredditData.csv', 'w') as newFile:
    writer = csv.writer(newFile)
    for k,v in subredditHourDictionary.items():
```

```
    writer.writerow([k,v])
```

Code Block F (\R)

#Install the R package arules

install.packages('arules');

```
#load the arules package
library("arules");
txn = read.transactions("comp_year.csv", format = "basket",rm.duplicates= FALSE ,sep=",", skip = 0)

# Run the apriori algorithm
basket_rules <- apriori(txn,parameter = list(minlen= 2,maxlen=5,sup = 0.01, conf = 0.01,target='rules'));

# Check the generated rules using inspect
inspect(basket_rules)

write(basket_rules,
     file = "comp_years_assoc.csv",
     sep = ",",
     quote = TRUE,
     row.names = FALSE)
```

Code Block G (Python)

```
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.cm as cm
import os
from datetime import datetime,timedelta
from random import random

files = [file for file in os.listdir('.') if '.pic' in file]
print(files)

subj_enroll_begin = {}
subj_enroll_end = {}
FMT = '%H%M'

fig = plt.figure()
ax1 = fig.add_subplot(111)

subjs = set()
for file_ in files:
        with open(file_,'rb') as file:
                contents = pickle.loads(file.read())
                for list_ in contents:
                        for subj in list_:
```

```
                                        for section in list_[subj]['data']:
                                                meetings = section['meetingsFaculty']
                                                subjs.add(section['subject'])
colors = cm.rainbow(np.linspace(0, 1,len(subjs)+1))
subjs = list(subjs)
legends = []
for file_ in files:
        with open(file_,'rb') as file:
                contents = pickle.loads(file.read())
                for list_ in contents:
                        for subj in list_:
                                for section in list_[subj]['data']:
                                        meetings = section['meetingsFaculty']
                                        enroll = section['enrollment']
                                        subject = section['subject']
                                        for meeting in meetings:
                                                begin = meeting.get('meetingTime').get('beginTime')
                                                end = meeting.get('meetingTime').get('endTime')
                                                if None in [begin,end]: continue
                                                clr = colors[subjs.index(subject)]
                                                tdelta = datetime.strptime(end, FMT) - datetime.strptime(begin, FMT)
                                                seconds_ = tdelta.total_seconds()/60 + random()
                                                ax1.scatter(enroll,seconds_,color=clr)
        print(file_)
subjs_set = set(subjs)
for sub in subjs_set:

        clr = colors[subjs.index(sub)]
        legend = mpatches.Patch(color=clr, label=sub) #label
        legends.append(legend)

plt.legend(handles=legends)
plt.show()
```

Figure A1

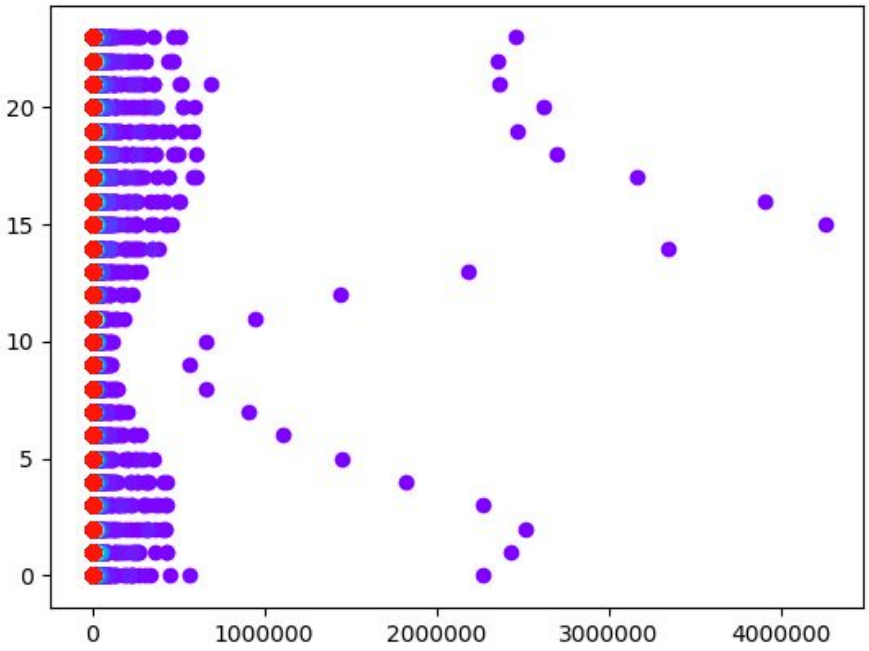

Figure A2

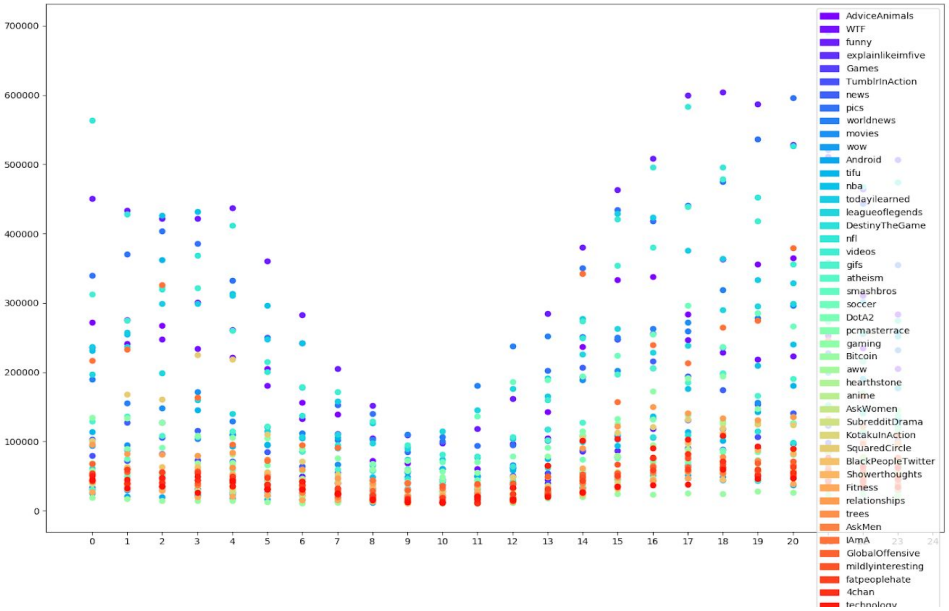

Figure A3

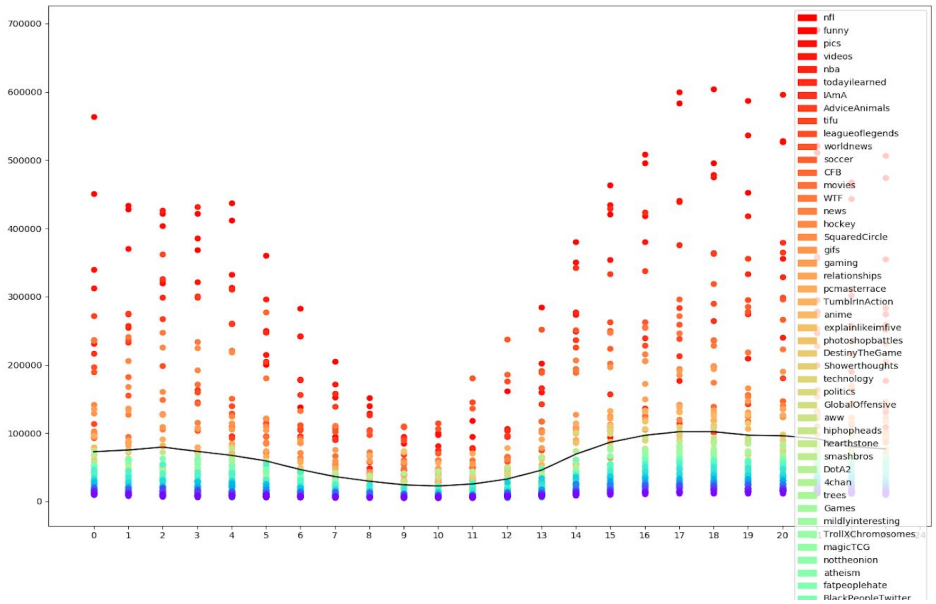

Figure A4

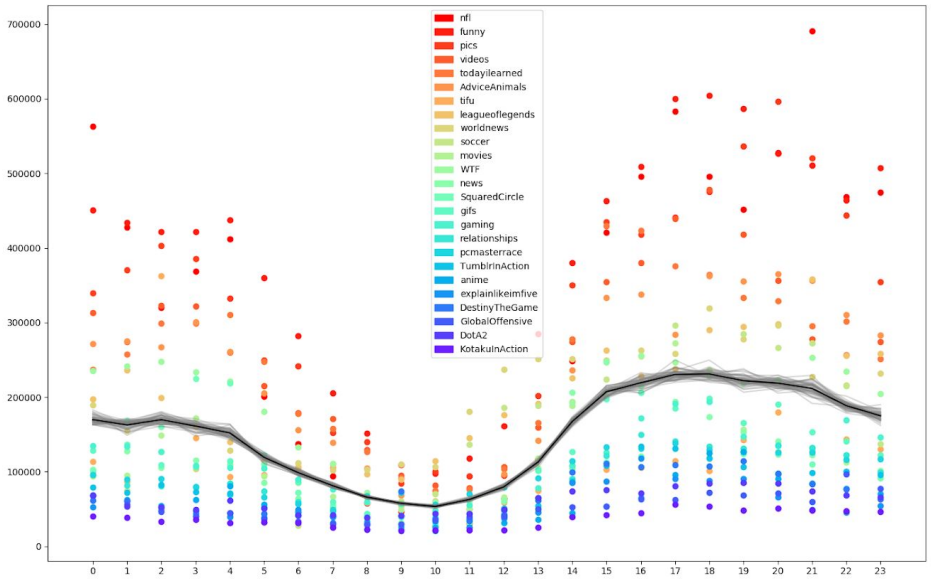

Figure A5

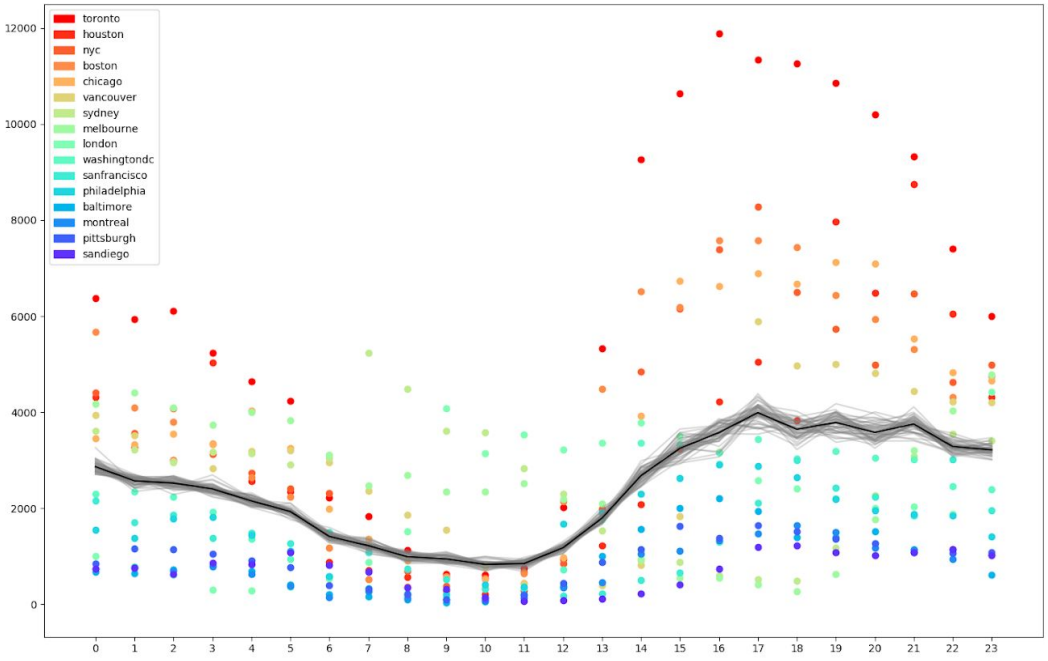

Figure A6

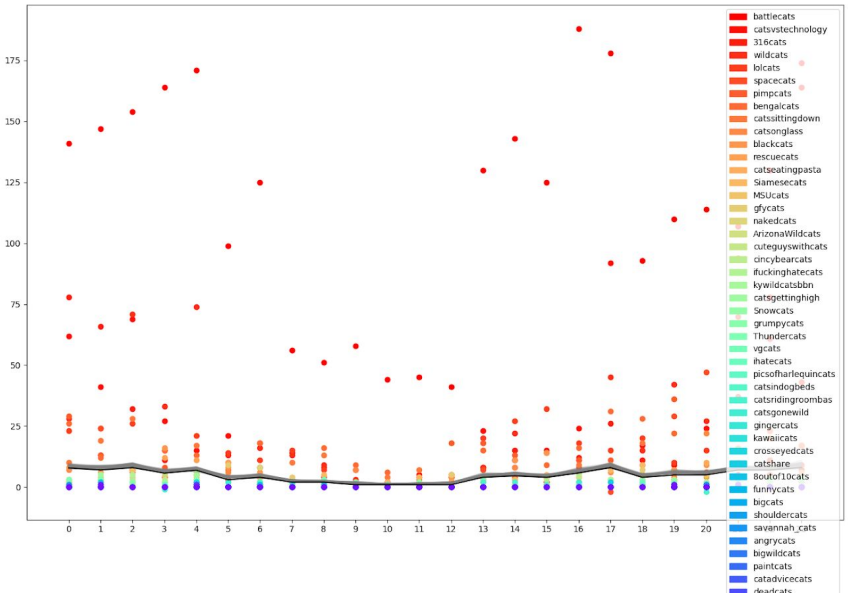

Figure A7

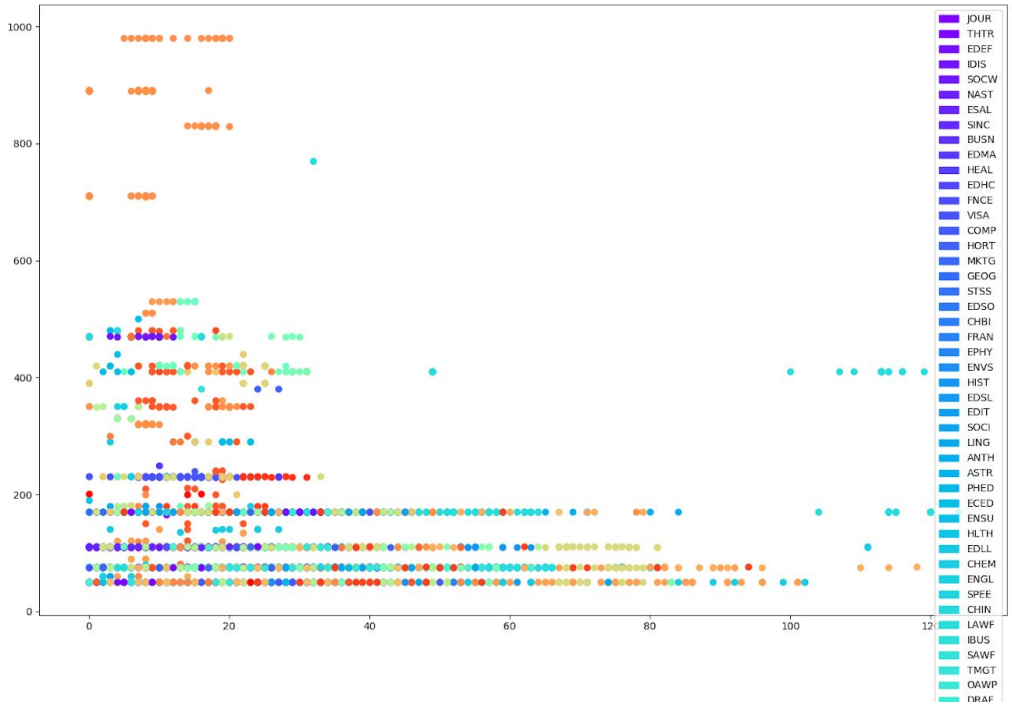

Figure B1

913	{4chan} => {pcmasterrace}	0.001635	0.182039	5.899314	3600
914	{pcmasterrace} => {4chan}	0.001635	0.052975	5.899314	3600
915	{4chan} => {technology}	0.001178	0.13122	5.878068	2595
916	{technology} => {4chan}	0.001178	0.052785	5.878068	2595
917	{4chan} => {WTF}	0.002794	0.311084	5.638125	6152
918	{WTF} => {4chan}	0.002794	0.05063	5.638125	6152
919	{4chan} => {AdviceAnimals}	0.002578	0.287116	4.516902	5678
920	{AdviceAnimals} => {4chan}	0.002578	0.040561	4.516902	5678
921	{Android} => {pcmasterrace}	0.0015	0.129401	4.19349	3304
922	{pcmasterrace} => {Android}	0.0015	0.04862	4.19349	3304
923	{Android} => {technology}	0.00202	0.174206	7.803665	4448
924	{technology} => {Android}	0.00202	0.090476	7.803665	4448
925	{Android} => {WTF}	0.001856	0.160067	2.901079	4087
926	{WTF} => {Android}	0.001856	0.033635	2.901079	4087
927	{Android} => {AdviceAnimals}	0.001979	0.170681	2.685153	4358
928	{AdviceAnimals} => {Android}	0.001979	0.031132	2.685153	4358
929	{millionairemakers} => {league	0.001254	0.052234	1.077067	2762
930	{leagueoflegends} => {milliona	0.001254	0.025861	1.077067	2762
931	{millionairemakers} => {nfl}	0.001267	0.052783	1.989606	2791
932	{nfl} => {millionairemakers}	0.001267	0.047771	1.989606	2791
933	{millionairemakers} => {pcmast	0.00183	0.076215	2.469879	4030
934	{pcmasterrace} => {millionaire	0.00183	0.059303	2.469879	4030
935	{millionairemakers} => {techno	0.001165	0.048528	2.17383	2566
936	{technology} => {millionairema	0.001165	0.052195	2.17383	2566
937	{millionairemakers} => {WTF}	0.002402	0.100043	1.8132	5290
938	{WTF} => {millionairemakers}	0.002402	0.043536	1.8132	5290

Figure B2

rules	support	confidence	lift	count
{nintendo} => {3DS}	0.001060283	0.296771734	56.49753397	2335
{3DS} => {nintendo}	0.001060283	0.201849931	56.49753397	2335
{gonewildcurvy} => {gonewild}	0.001191058	0.394970637	37.5684067	2623
{gonewild} => {gonewildcurvy}	0.001191058	0.113289854	37.5684067	2623
{CollegeBasketball} => {CFB}	0.001380865	0.422537168	36.63357825	3041
{CFB} => {CollegeBasketball}	0.001380865	0.119719696	36.63357825	3041
{halo} => {xboxone}	0.001042119	0.236939913	31.39964288	2295
{xboxone} => {halo}	0.001042119	0.138103262	31.39964288	2295
{Gunners} => {soccer}	0.001143834	0.47519336	31.19663885	2519
{soccer} => {Gunners}	0.001143834	0.075093158	31.19663885	2519
{AskWomen} => {AskMen}	0.002104218	0.276673234	29.47615952	4634
{AskMen} => {AskWomen}	0.002104218	0.224178801	29.47615952	4634
{reddevils} => {soccer}	0.001009425	0.444955965	29.21154147	2223
{soccer} => {reddevils}	0.001009425	0.066269191	29.21154147	2223
{atheism,technology} => {politics}	0.001179706	0.471164309	27.58229356	2598
{nba,nfl} => {CFB}	0.001260533	0.311420238	26.99984405	2776
{CFB,nba} => {nfl}	0.001260533	0.693826543	26.15320157	2776
{KotakuInAction} => {TumblrInAction}	0.001003068	0.258665105	25.47942111	2209
{TumblrInAction} => {KotakuInAction}	0.001003068	0.098805743	25.47942111	2209
{cowboys} => {nfl}	0.001161543	0.591718714	22.30433376	2558
{nfl} => {cowboys}	0.001161543	0.043783377	22.30433376	2558
{csgobetting} => {GlobalOffensive}	0.002243622	0.401642009	22.27039564	4941
{GlobalOffensive} => {csgobetting}	0.002243622	0.124405167	22.27039564	4941
{GlobalOffensiveTrade} => {GlobalOffensive}	0.001488028	0.395820751	21.94761636	3277
{GlobalOffensive} => {GlobalOffensiveTrade}	0.001488028	0.082508749	21.94761636	3277
{Patriots} => {nfl}	0.002031565	0.572855314	21.59329393	4474
{nfl} => {Patriots}	0.002031565	0.076578119	21.59329393	4474
{AdviceAnimals,technology,WTF} => {politics}	0.00124464	0.263225101	21.26080506	2741